(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
12 April 2001 (12.04.2001)

PCT

(10) International Publication Number
WO 01/25932 A1

(51) International Patent Classification⁷: G06F 12/14, 12/16, H04L 9/00

(21) International Application Number: PCT/US00/26896

(22) International Filing Date:
29 September 2000 (29.09.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:

60/157,472	1 October 1999 (01.10.1999)	US
60/206,947	25 May 2000 (25.05.2000)	US

(71) Applicant *(for all designated States except US)*: **INFRAWORKS CORPORATION** [US/US]; 504 Lavaca Street, Suite 1100, Austin, TX 78701 (US).

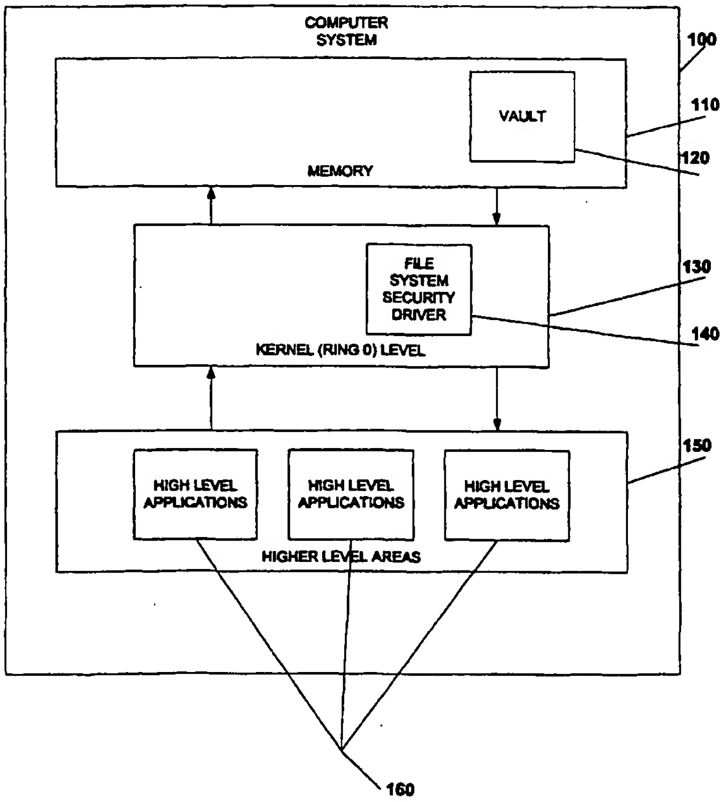
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **FRIEDMAN, George** [US/US]; 7109 Montana Norte, Austin, TX 78731 (US). **STAREK, Robert, Phillip** [US/US]; 3609 Del Robles, Austin, TX 78727 (US). **MURDOCK, Carlos** [US/US]; 4517 Avenue F, Austin, TX 78751 (US).

(74) Agents: **TAUFER, Paul, A.**; Schnader Harrison Segal & Lewis LLP, 1600 Market Street - Suite 3600, Philadelphia, PA 19103-7286 et al. (US).

(81) Designated States *(national)*: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

[Continued on next page]

(54) Title: BACK-CHANNELING IN A MEMORY VAULT SYSTEM



(57) Abstract: A file system security driver and vault method and system particularly applicable to a system in which protected data is segregated from other data, which allows for back-channeling of file data in order to ensure that files created by applications using secured data do not cause data leaks of secure data. In a preferred embodiment, a file system security driver (140) is a driver resident on the kernel level (130) which monitors files system requests and allows limited access to files resident on the vault (120) and creation of files within the vault (120) when necessary.

(84) **Designated States** *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:

— *With international search report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

BACK-CHANNELING IN A MEMORY VAULT SYSTEM

Field of the Invention:

The invention relates to the protection of data stored in a computer; and more particularly to the protection of data stored in a memory vault system.

Background of the Invention:

In order to secure data from unauthorized access, the data may be encrypted. Encryption algorithms are commonly designed to use a key pair – one key for encryption and one key for decryption. Encryption is used to encode information and send it as an encrypted file. Encryption may also be used to set up a secure connection between two computers so that eavesdroppers will not be able to intercept data being exchanged.

An encrypted container may be used to protect data within a computer system. An encrypted container is a reserved area on a disk to which protected files are copied and saved. This may be implemented by creating a file which is encrypted, and which is mounted as and acts as a file system. Prior art encrypted containers allow content distributors to distribute content to users without endangering the security of the content. A client side container-opener application is used to access the encrypted container. The client side container-opener may limit the accesses to data in the encrypted container in any number of ways. For example, it may allow the data to be accessed only at certain times or with a proof of payment key.

Prior art encrypted container systems contain limited security once the container is opened and the data released. When a file is opened in a computer running prior art software and operating systems, the data may leak to applications (such as clipboard or other system applications) and may be copied and left insecure during system operations, such as printing. This compromises the system.

Some client side container-openers may come with integrated data-display mechanisms which allow the user to view the secure data. For example, Folio (NextPage, Inc.) provides a viewer browser to display files and the viewer browser prevents unauthorized use of data by turning off application level controls. However, this means that when documents in Folio are displayed, data can still be saved to the clip board and attacked in other ways from the system level.

Watermarking or digital fingerprinting by the client side container-opener application can be used to trace the origin of content which has been opened and removed from an encrypted container. This will allow tracing of content which has been distributed in an unauthorized way. However, this still allows the data to be viewed by unauthorized users.

Summary of the Invention:

The invention discloses a memory vault system and method particularly applicable to a system in which protected data is transmitted to a recipient with access controls. An illustrative embodiment of the invention comprises a computer system in which secured data in a memory vault is accessed via a system-level security application which enforces strict access controls on data, and outside applications permitted to access data are monitored and certain system-level commands from these outside applications mediated by the security application. Back-channeling of any data derived from secured data is effected so that all data remains secured.

Description of the Drawings:

The invention is best understood from the following detailed description when read with the accompanying figures.

Figure 1 is an schematic diagram of a computer system operating according to an illustrative embodiment of the back-channeling method of the invention.

Figure 2 is a flow chart of a file open in a memory vault system according to an illustrative embodiment of the back-channeling method of the invention.

Figure 3 is a flow chart of a file read/write in a memory vault system according to an illustrative embodiment of the back-channeling method of the invention.

Figure 4 is a flow chart of a file information request in a memory vault system according to an illustrative embodiment of the back-channeling method of the invention.

Figure 5 is a flow chart of a file change request in a memory vault system according to an illustrative embodiment of the back-channeling method of the invention.

Detailed Description of the Invention:

The invention disclosed allows a computer system including a memory vault system to include the use of secured data from the vault by insecure applications without modification of these applications or security breaches. This is done by implementing back-channeling of data used or produced by these insecure applications (in other words, the segregation of this kind of data and the creation of any of this data in the vault rather than elsewhere in system memory) and thereby protecting in the same way that the original secured data is protected.

With reference to Figure 1, in a preferred embodiment, the back-channeling method is implemented on a computer system 100. Computer system 100 contains memory 110, which may be configured in any number of ways and may include several memory systems and kinds of memory media. Resident in memory 110 is at least one vault 120. In a preferred embodiment, a vault 120 uses the logical construct of a file, and within it is embedded a simple, robust file structure able to manage data from heterogenous file sources. Memory 110 and hardware are directly accessed only by kernel-mode applications resident in the kernel (ring 0) level 130. One such kernel-mode application is the file system security driver 140. Higher-level applications 160 which are in any higher level (collectively shown in Fig. 1 as 150) access memory only via applications resident in the kernel (ring 0) level 130, as shown by the arrows. In a preferred embodiment of the invention, the security system associated with file system security driver 140 ensures that requests for data by higher-level applications 160 are always handled by file system driver 140. The file system security driver 140 keeps information on secured processes running, file handles that higher-level applications use to reference secured files and other vault files, and vault file handles which file system security driver 140 can use to access secured files and other vault files. The file system security driver 140 uses this information to ensure that access to vault information is possible but restricted. It does this by intercepting and acting on file open, file read/write, file information, and file change requests. These requests are handled as shown in Figures 2 through 5.

With reference to Figure 2, a process sends a file open request which is received by file system security driver 140 (Fig. 1). Upon receipt of a file open request 200, the request is examined to see if it is for a secured file 205. If it is, then the request is examined to see if the requesting process is on the secured process list 210.

If the request is for a secured file, and the requesting process is not on the secured process list, then a security check is performed by a security driver on the requesting process 215. If the process does not pass the security check, then the requesting process gets access to a 0-byte tag file instead of the file requested 220. If the process does pass the security check, then permissions are checked, and the user is queried about whether the user wants to open a secured file 225. If permission is not allowed, or if the user does not want to open it, then the requesting process gets access to a 0-byte tag file instead of the file requested 220. If the user does want to open it, the process is added to the secured process list 230. (Information about the opening of a secured file will be passed to the part of the security driver in order to monitor the number of file opens permitted.) The process will be granted a file handle and a corresponding vault file handle will be created; these handles will be stored for indexing on the opened file list 235. In this way, a file is opened and ready for back-channeling 240.

If the request is for a secured file, and the requesting process is on the secured process list then if the process has not previously opened the requesting file the check and query 225 will occur. If the process has opened it before, the process activity is back channeled. The process is granted a file handle and a corresponding vault file handle will be created; these handles will be stored for indexing on the opened file list 235. A file is open and ready for back-channeling 240.

If the request is not for a secure file, then the secured process list is consulted to see if the request was made by a secure process 245. If it is, then a check is made to see if the file exists 250. If it does not, then in order to back-channel, a file is created in the vault which corresponds to the file request 255. In this way, a secured process is not allowed to create a file outside of the vault (an insecure file). This is back-channeling – not permitting a secured process to "leak" secure information to a file which is not protected by the vault.

If the request is by a secure process, but for a file which is not secure, then the file request flags for the file open request are modified so that no data in the insecure file can be erased or modified 260. In this way a secured process can access non-secure data outside of the vault, but can not write to it - preventing secure data from being written to an insecure (non-vault) file. The modified request is passed to the file system for completion 265.

If the request is not for a secure file and not made by a secure process, then the request has no security implications, and the request is passed to the file system for completion 265.

In this way, secured data can only be accessed by secured processes, new files opened by secured processes are always created in the vault, and insecure files opened by secured processes can not be written to.

As shown in Fig. 3, back-channeling is also enforced when file read/write requests are intercepted by the file system security driver (140, Fig. 1). When such a request is received 300, a check is performed to see if the file handle is on the opened file list 305. If it is, then the file is in the vault. A check is performed to see if the requesting process is on the secured process list 310. If it is, then the request is completed by performing the request on the corresponding vault file 315, thus back-channeling the read or write request 320. If it is not, then the request is for a secured file by an insecure process, and the request is denied 325.

If the file handle was not on the opened file list at check 305, a further check is done to see if the process is on the secured process list 330. If it is not, then the file system is allowed to complete the request 335. If it is, then the request is examined to see if it is a write request 340. If it is not, the file can be opened, even though it is insecure and the process is secure, since no secure data can be leaked by reading an insecure file. The request is passed to the file system 335. If the file is insecure, the process secure, and the request is a write - in this situation there is the danger of a data leak. Therefore the write request is blocked 325.

File information requests are handled differently - the correct file size can be accessed even by insecure processes. This is file size spoofing -- in the preferred embodiment, the file system sees the secured file's reference as pointing to a zero-byte file, and if a file-size request were handled by the file system, it would return a size of zero. In the preferred embodiment, non-secure processes are allowed to see the size of secure files, and therefore, with reference to Figure 4, when the request is received 400, a check is performed to see if it is a vault file 405. If it is not a vault file, then the request is passed to the file system to complete 410. If it is a handle for a vault file or the name of the vault file, then size of the corresponding actual vault file is checked and returned 415.

File change requests - change requests which are not reads, writes, or opens - are handled similarly to file writes. With reference to Figure 5, when a file change request is received 500, the request is examined to see if the file referenced is a vault file 505. If it is, then the opened file list and other databases are updated with new file information 510. If the file is not a vault file, then the process is checked to see if it is a secured process 515. If it is not a secured process, then no security concern at all is implicated, and the request is passed to the file system for completion 520. If it is a secured process but the file is not secured, then the request is blocked, to ensure that the secured process will not corrupt previously existing data 525.

A further illustrative embodiment of the invention is directed to a file system security driver wherein the driver implements a vault system with back-channeling according to the methods provided herein. The illustrative file system security driver comprises a driver resident on the kernel level which monitors file system requests and allows limited access to files resident on the vault and creation of vault files when necessary.

Further disclosed is a secured data transmission system having a receiver component to access secured file content provided by a sender, wherein the receiver includes a vault system and a file system security driver acts according to the methods provided herein.

Still further disclosed is a computer configured to include a vault system and a computer-readable medium programmed to monitor file system requests and allow limited access to files resident on the vault and creation of vault files, according to the methods provided herein.

The terms "computer", "computer system", or “system” as used herein include any electronic device having a processor or microprocessor including, without limitation, a personal computer, such as a laptop, palm PC, desktop or workstation, a network server, a mainframe, an electronic wired or wireless device, such as for example, a telephone, an interactive television, such as for example, a television adapted to be connected to the Internet or an electronic device adapted for use with a television, a cellular telephone, a personal digital assistant, an electronic pager, a digital watch, or any other device capable of receiving information, such as email, from another source. A computer, computer system, or system of the invention may operate in communication with other systems over a network, such as, for example, the Internet, an intranet, or an extranet, or may operate as a stand-alone system.

While the invention has been described by illustrative embodiments, additional advantages and modifications will occur to those skilled in the art. Therefore the invention in its broader aspects is not limited to specific details shown and described herein. Modifications may be made without departing from the spirit and scope of the invention. Accordingly, it is intended that the invention not be limited to the specific illustrative embodiments but be interpreted within the full spirit and scope of the appended claims and their equivalents.

[I / We] claim:

1. A method for securing data comprising:

a vault provision step of providing a vault system for segregating vault data from other system data; and

a file system security driver provision step of providing a file system security driver which intercepts file system calls, and for each specific one of said intercepted file system calls, determines whether said specific one of said intercepted file system calls is from a process accessing said vault data, and, if said specific one of said intercepted file system calls is from a process accessing said vault data, permitting the file system call to create or modify data only within said vault system.

2. The method of claim 1, where said file system security driver provision step further comprises a file open handling step of, for each specific one of said intercepted file system calls which is a file open call, comprising the steps of:

determining whether said file open call is a request for data from among said vault data; and

if said file open call is a request for data from among said vault data, performing a check on process making said request to see if said process is already a secured process which has previously opened said data from among said vault data, and if so, allowing access to said vault data, and if not performing an access check on process making said request, and then processing the request by allowing access to said process which is not already a secured process if said access check is passed but not allowing access at all if said access check is not passed;

if said file open call is not a request for data from among said vault data, performing a check on said process making said request to see if said process is already a secured process, and passing the request onto an operating system if said process making said request is not a secured process, and, if said process making said request is a secured process, determining if file referred to in said file open call exists, and if it does, opening said file for read only, and if it does not, creating said file in said vault data.

3. The method of claim 2, where said processing the request by allowing access to said process which had not previously been granted access to said vault data comprises the step of:

querying user to determine if said user would like to open said data from among said vault data, and opening said data from among said vault data only if said user would like to open said data.

4. The method of claim 3, where said processing the request by allowing access to said process which had not previously been granted access to said vault data comprises the step of:

recording said allowed access and monitoring total accesses allowed.

5. The method of claim 2, where said processing the request by allowing access to said process which had not previously been granted access to said vault data comprises the step of:

recording said process which had not previously been granted access to said vault data making said request in a list of processes allowed to access said vault data.

6. The method of claim 2, where said step of creating said file in said vault data comprises the step of:

sending said secured process a stand-in file handle;

creating a corresponding vault file handle; and

storing said stand-in file handle and said corresponding vault file handle.

7. The method of claim 2, where said step of opening said file for read only comprises the

steps of:

modifying any file request flags of said file open call which indicating modification of the file is permitted; and

passing said modified file open call to said operating system.

8. The method of claim 1, where said file system security driver provision step further comprises a file read/write request handling step of, for each specific one of said intercepted file system calls which is a file read/write call, comprising the steps of:

determining whether said read/write request is a request for data from among said vault data;

if said read/write request is a request for data from among said vault data, allowing access if process making said request is allowed to access said vault data; and

if said read/write request is a request for data not from among said vault data, allowing access if said process making said request is not allowed to access said vault data, and allowing access if said read/write request is a read request.

9. The method of claim 1, where said file system security driver provision step further comprises a file information request step, comprising the step of:

determining whether said file information request is a request regarding data from among said vault data, and if not, passing said file information request to said operating system, and if so, discerning correct file size and returning said correct file size.

10. The method of claim 1, where said file system security driver provision step further comprises a file change request step, comprising the step of:

determining whether said file change request is a request regarding data from among said vault data, and if so performing said file change request on said vault data, and if not, checking to see if the requesting process is a secured process, and if not, passing said file change request to said operating system, and if so, blocking the request.

11. The method of claim 1, where said file system security driver provision step further comprises:

a file open handling step of, for each specific one of said intercepted file system calls which is a file open call, comprising the steps of:

determining whether said file open call is a request for data from among said vault data; and

if said file open call is a request for data from among said vault data, performing a check on process making said request to see if said process is already a secured process which has previously opened said data from among said vault data, and if so, allowing access to said vault data, and if not performing an access check on process making said request, and then processing the request by allowing access to said process which is not already a secured process if said access check is passed but not allowing access at all if said access check is not passed; and

if said file open call is not a request for data from among said vault data, performing a access check on said process making said request to see if said process is already a secured process, and passing the request onto an operating system if said process making said request is not a secured process, and, if said process making said request is a secured process, determining if file referred to in said file open call exists, and if it does, opening said file for read only, and if it does not, creating said file in said vault data;

a file read/write request handling step of, for each specific one of said intercepted file system calls which is a file read/write call, comprising the steps of:

determining whether said read/write request is a request for data from among said vault data;

if said read/write request is a request for data from among said vault data, allowing access if process making said request is allowed to access said vault data; and

if said read/write request is a request for data not from among said vault data, allowing access if said process making said request is not allowed to access said vault data, and allowing access if said read/write request is a read request;

a file information request step, comprising the step of:

determining whether said file information request is a request regarding data from among said vault data, and if not, passing said file information request to said operating system, and if so, discerning correct file size and returning said correct file size; and

a file change request step, comprising the steps of:

determining whether said file change request is a request regarding data from among said vault data, and if so performing said file change request on said vault data, and if not, checking to see if the requesting process is a secured process, and if not, passing said file change request to said operating system, and if so, blocking the request.

12. A file system security driver and vault system wherein said vault system segregates vault data from other system data; and said file system security driver intercepts file system calls, and for each specific one of said intercepted file system calls, determines whether said specific one of said intercepted file system calls is from a process accessing said vault data, and, if said specific one of said intercepted file system calls is from a process accessing said

vault data, permitting the file system call to create or modify data only within said vault system.

13. A file system security driver and vault system wherein said vault system segregates vault data from other system data; and said file system security driver intercepts file system calls, and for each specific one of said intercepted file system calls, determines whether said specific one of said intercepted file system calls is from a process accessing said vault data, and, if said specific one of said intercepted file system calls is from a process accessing said vault data, permitting the file system call to create or modify data only within said vault system.

14. A secured data transmission system having a receiver to access secured file content provided by a sender, wherein the receiver includes a file system security driver and vault system and where said vault system segregates vault data from other system data; and said file system security driver intercepts file system calls, and for each specific one of said intercepted file system calls, determines whether said specific one of said intercepted file system calls is from a process accessing said vault data, and, if said specific one of said intercepted file system calls is from a process accessing said vault data, permitting the file system call to create or modify data only within said vault system.

15. A computer configured to protect secure data by including a file system security driver and vault system and where said vault system segregates vault data from other system data; and said file system security driver intercepts file system calls, and for each specific one of said intercepted file system calls, determines whether said specific one of said intercepted file system calls is from a process accessing said vault data, and, if said specific one of said intercepted file system calls is from a process accessing said vault data, permitting the file system call to create or modify data only within said vault system.

16. A computer-readable medium programmed to protect secure data by including a file system security driver and vault system and where said vault system segregates vault data from other system data; and said file system security driver intercepts file system calls, and for each specific one of said intercepted file system calls, determines whether said specific one

of said intercepted file system calls is from a process accessing said vault data, and, if said specific one of said intercepted file system calls is from a process accessing said vault data, permitting the file system call to create or modify data only within said vault system.

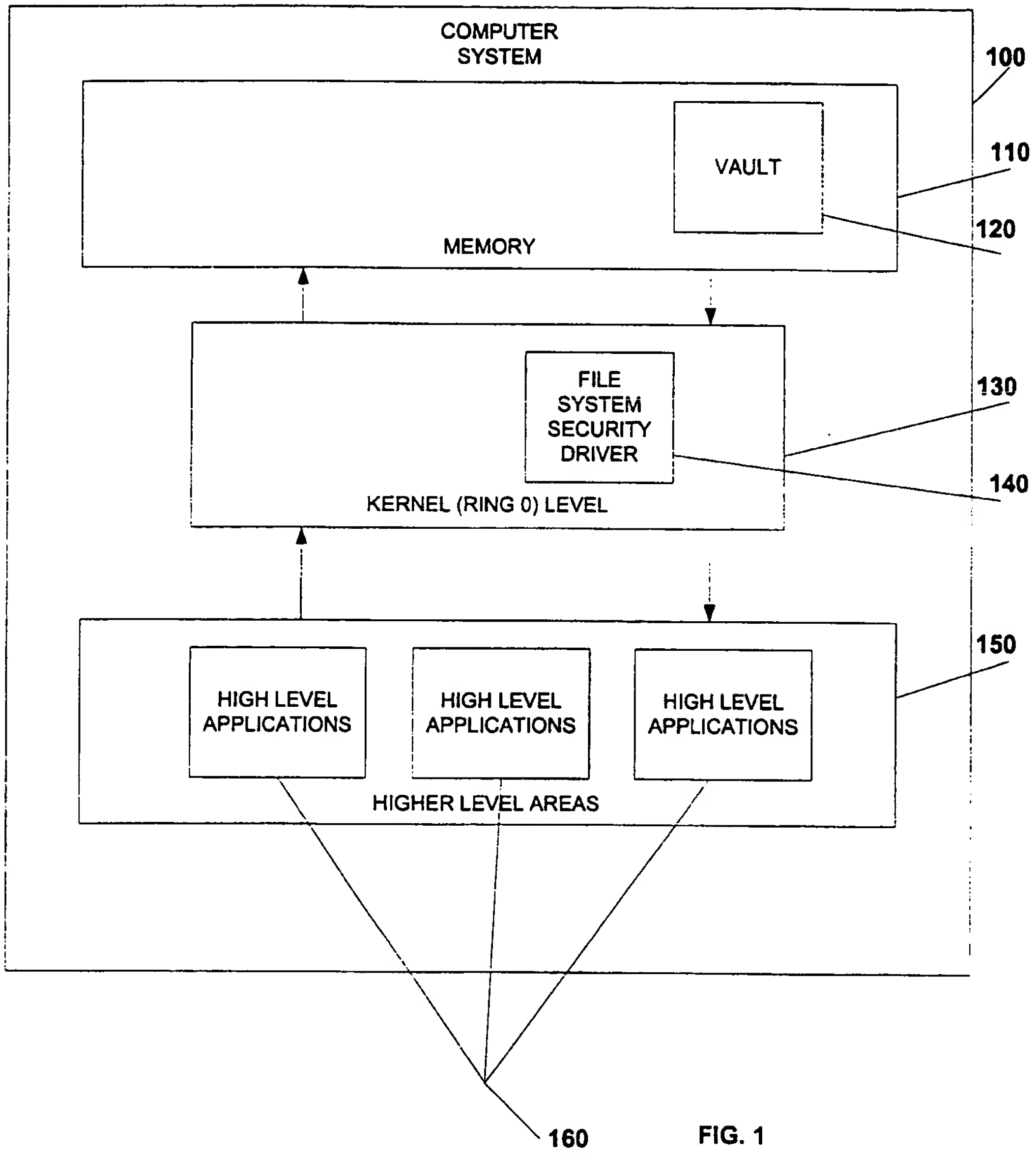

COMPUTER SYSTEM
100
VAULT
110
120
MEMORY
FILE SYSTEM SECURITY DRIVER
130
140
KERNEL (RING 0) LEVEL
HIGH LEVEL APPLICATIONS
HIGH LEVEL APPLICATIONS
HIGH LEVEL APPLICATIONS
150
HIGHER LEVEL AREAS
160

FIG. 1

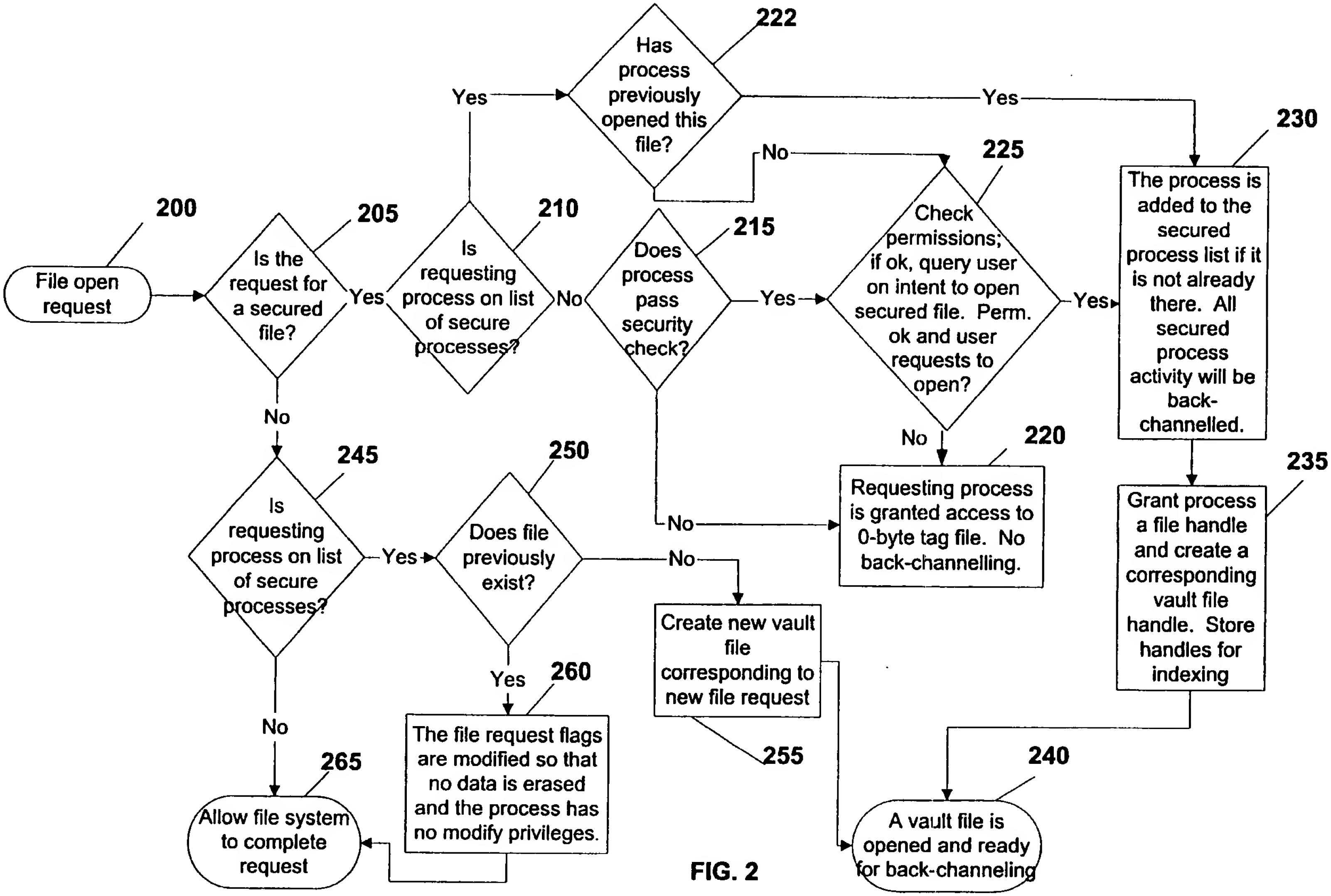

200
File open request
205
Is the request for a secured file?
Yes
No
210
Is requesting process on list of secure processes?
Yes
No
222
Has process previously opened this file?
Yes
No
215
Does process pass security check?
Yes
No
225
Check permissions; if ok, query user on intent to open secured file. Perm. ok and user requests to open?
Yes
No
230
The process is added to the secured process list if it is not already there. All secured process activity will be back-channelled.
235
Grant process a file handle and create a corresponding vault file handle. Store handles for indexing
220
Requesting process is granted access to 0-byte tag file. No back-channelling.
240
A vault file is opened and ready for back-channeling
245
Is requesting process on list of secure processes?
Yes
No
250
Does file previously exist?
Yes
No
255
Create new vault file corresponding to new file request
260
The file request flags are modified so that no data is erased and the process has no modify privileges.
265
Allow file system to complete request

FIG. 2

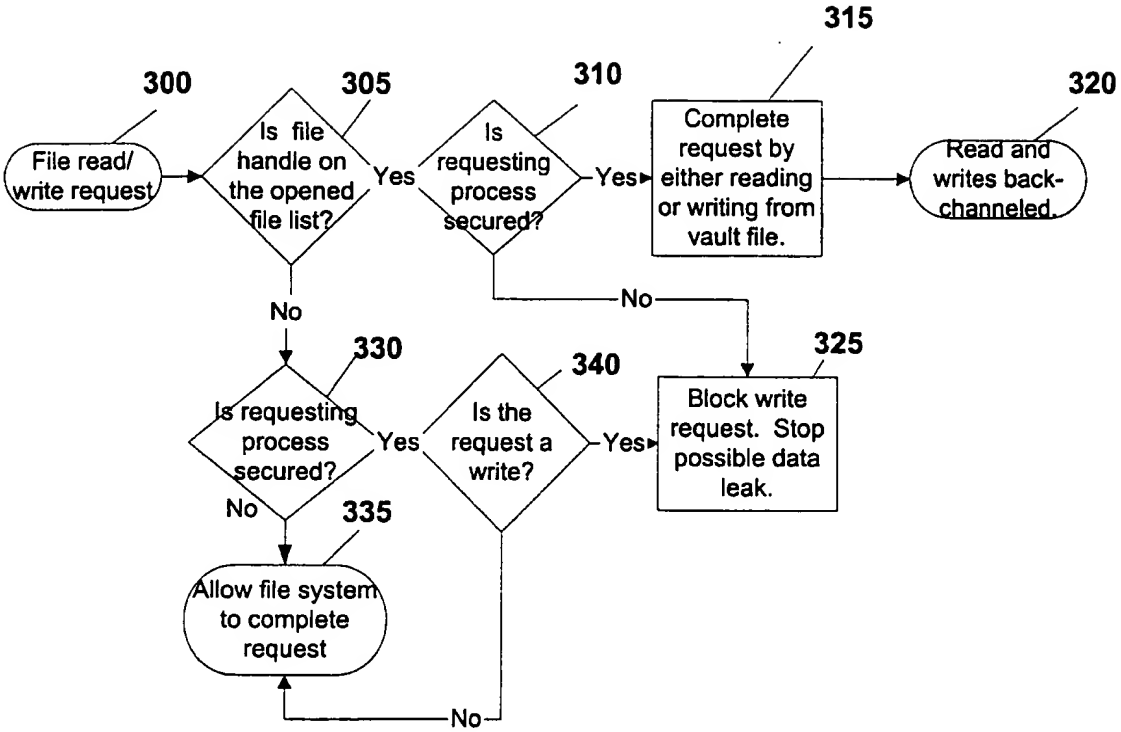

300
File read/ write request
305
Is file handle on the opened file list?
Yes
310
Is requesting process secured?
Yes
315
Complete request by either reading or writing from vault file.
320
Read and writes back-channeled.
No
No
325
Block write request. Stop possible data leak.
330
Is requesting process secured?
Yes
340
Is the request a write?
Yes
No
335
Allow file system to complete request
No

FIG. 3

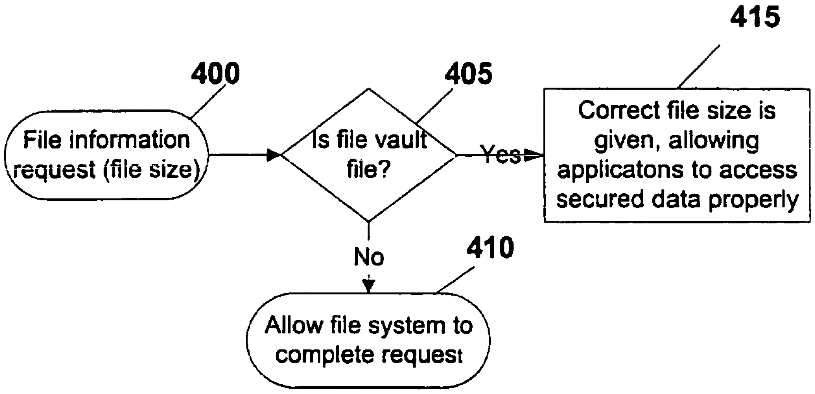

400
File information request (file size)
405
Is file vault file?
Yes
415
Correct file size is given, allowing applicatons to access secured data properly
No
410
Allow file system to complete request

FIG. 4

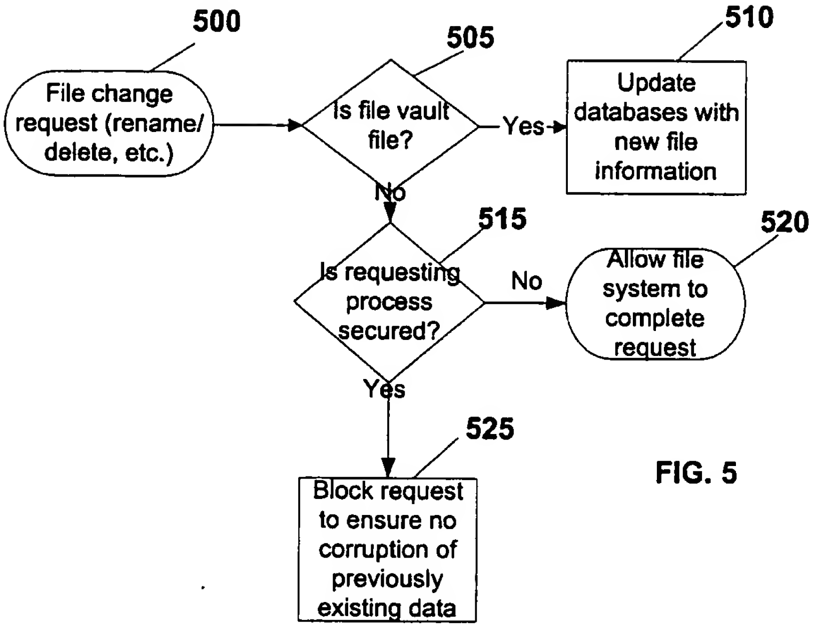

500
File change request (rename/ delete, etc.)
505
Is file vault file?
Yes
510
Update databases with new file information
No
515
Is requesting process secured?
No
520
Allow file system to complete request
Yes
525
Block request to ensure no corruption of previously existing data

FIG. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/26896

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 12/14. 12/16; H04L 9/00
US CL :713/200. 164. 165. 166; 707/1. 9; 710/36. 200; 711/100. 163
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 713/200. 164, 165, 166; 707/1. 9; 710/36, 200; 711/100, 163

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and. where practicable. search terms used)

STN, EAST
search terms: security, kemel. access. (file or data or information)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,347,578 A (DUXBURY) 13 September 1994, Abstract.	1, 12, 13, 14, 15 and 16
A	US 4,621,321 A (BOEBERT et al.) 04 November 1986, Claims 7 and 13.	1, 12, 13, 14, 15 and 16.
A	US 5,956,710 A (YAROM) 21 September 1999, Figures 1 and 4.	1, 12, 13, 14, 15 and 16.

□ Further documents are listed in the continuation of Box C. □ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance. the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date		
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed	"&"	document member of the same patent family

Date of the actual completion of the international search 29 DECEMBER 2000	Date of mailing of the international search report 22 JAN 2001
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer LY V. HUA Peggy Harrod Telephone No. (703) 305-9684